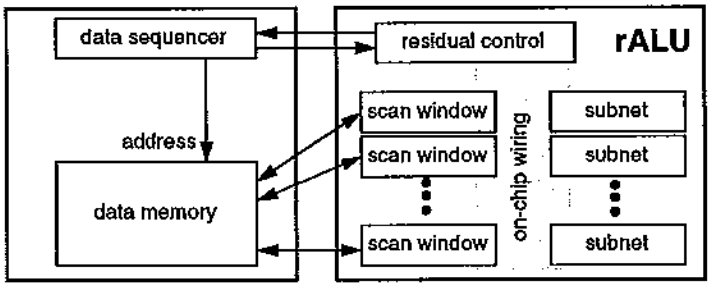
rALU

rALU Architectures for the Xputer Prototype MoM-3

Rainer Kress

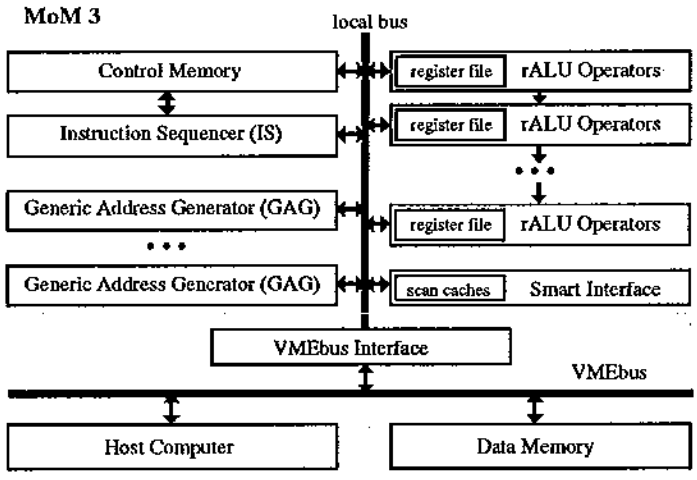
- Introduction
- rALU Emulator
- FPGA based rALU
- Data-Driven rALU
- Conclusions

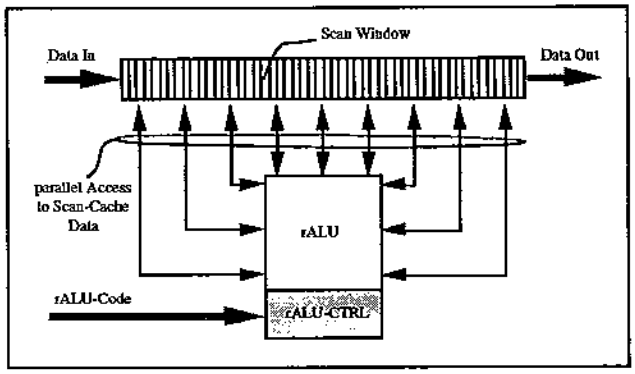


• Xputer Architecture

- Focus on rALU hardware including:
  - rALU subnet
  - scan windows
  - residual control
- Focus on programming the different rALU architectures

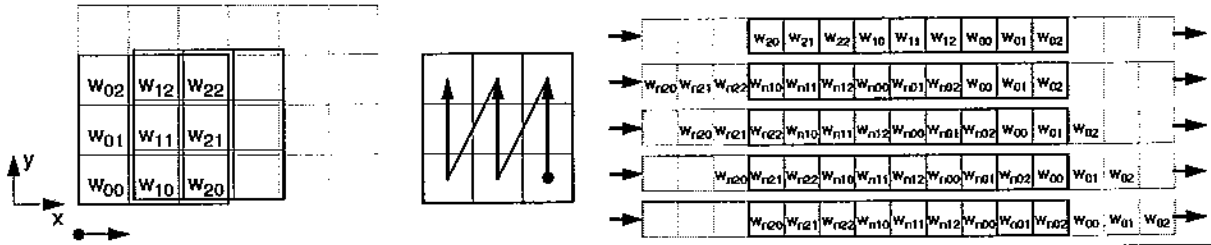
• MoM-3 Architecture

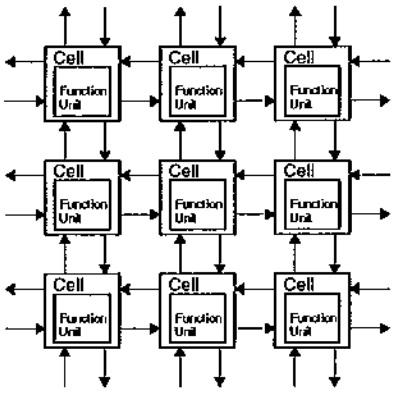




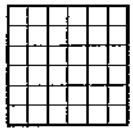
• rALU-Emulator

- emulation with 68020
- 1024 bit wide shift-register as scan window
- reprogrammable
- one operation at a time
- an optimized read/write strategy can be used

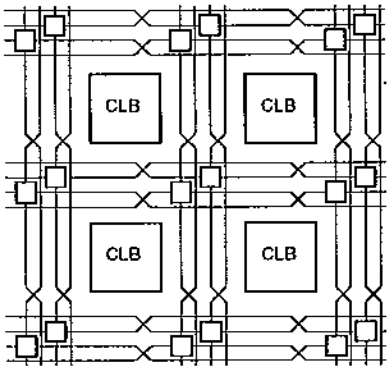




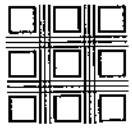
• Algotronix CAL 1024



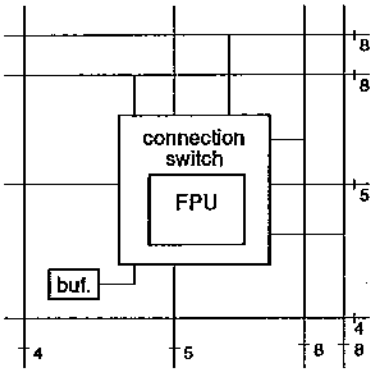
- sea-of-gates structure, fine grained
- Function Unit: - 2 input 1 output comb. logic function
  - 4 kinds of D-Latches
- 2 global signals to all cells without passing multiplexers (e. g. for clocks) and nearest neighbour connections
- 32 x 32 cell design, 144 pins, 245 pounds per chip
- example implementation: counter 50 MHz



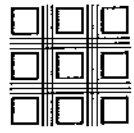
• XILINX XC4000 (XC3000, XC2000)



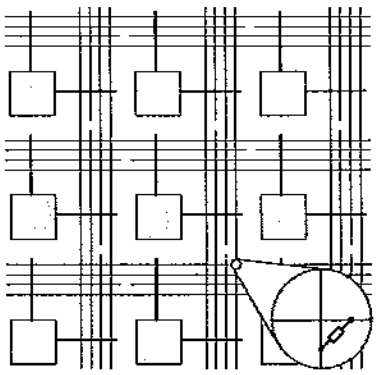
- symmetrical array structure, coarse grained
- Function Unit: - 2 independent 4 input FUs
  - 2 output comb. 2 registered
- single-length, double-length, longlines, all metal
- fast arithmetic carry, FU can be used as RAM, wide decoders
- 18 x 18 (30 x 30, XC4020) CLB matrix, 936 (2280) flip-flops, 144 (240) IOBs, 1200 DM per Chip
- example implementation: 16-bit Adder/Subtractor 38 MHz, 9 CLBs; 16-bit loadable counter 30 MHz, 16 CLBs



• AT&T: Optimized Reconfigurable Cell Array (ORCA)



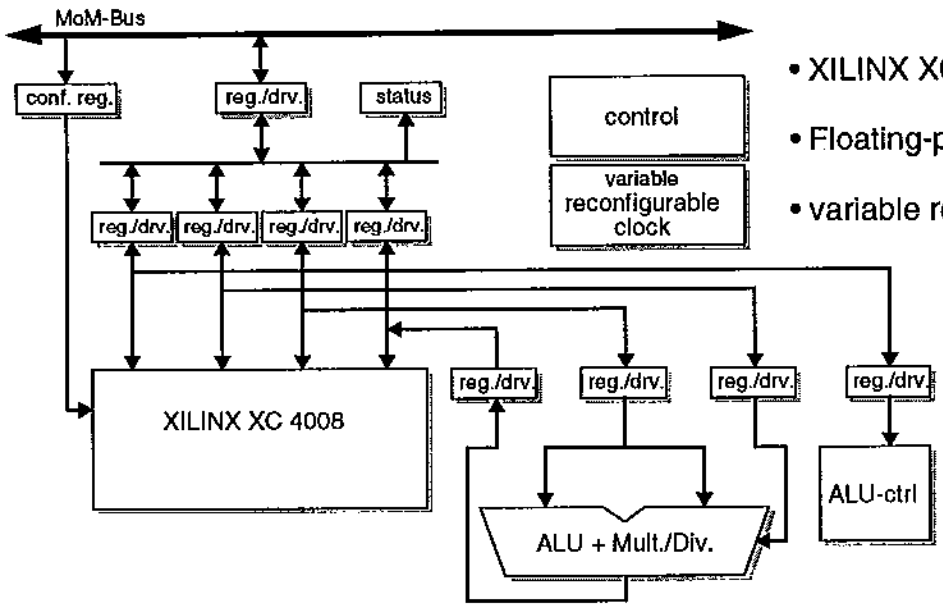
- symmetrical array structure, coarse grained
- Function Unit: - 4 x 4 or 2x 5 or 1x 6 input FUs
  - 5 output, 4 registered
- support for large data manipulations (a 16 bit adder requires only 4 blocks)
- support for wide data registers
- support for flexible, on-chip application memories (16 x 4 memory in one cell)
- fast carry circuit
- direct mapping of XC3000 to ORCA



• APTIX AX1024D / AX1024R

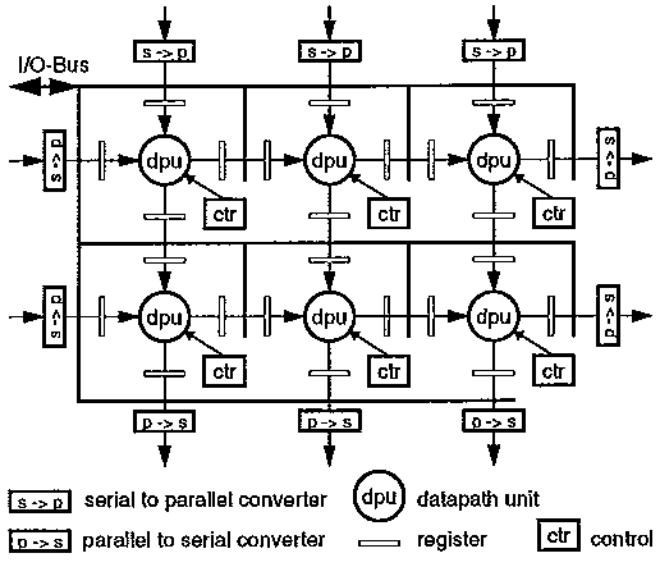
- Field Programmable Interconnect Components (FPIC)
- AX1024D with 64 diagnostic pins
- 32 x 32 pin matrix (1024 pins)
- bidirectional I/Os (120 Ω / 20 pF)
- 0.8 µ CMOS, over 1M transistors, signal range 0..5 V

Producer	Name/Parts	Programming Environment
IDA Supercomputing Research Center 1992	**SPLASH2** • 17x XC 4010 • 1x Crossbar per Modul • up to 16 Moduls • SBus	• Programming Models - SIMD - 1 dim. Pipelined Systolic Model - several higher dim. systolic Models • VHDL with several handmade tools • dBC (Data Parallel Bit Serial C) to VHDL
Virtual Computer Corporation 1992	**Virtual Computer** • 52x XC 4010 • 24x I-Cube IQ160 array • SBus	• only XILINX and I-Cube tools
GO Giga Operations Corporation 1993	**G200** • 8x 21xx DSPs • 14 MB DRAM • 7x XC4000 FPGAs connected together with bus • ISA Bus to 386/486 PC **G800** Motherboard • max. 8 computing modules DSPMOD or PGAMOD • VL Bus **DSPMOD** • 4x DSPs • 4 MB DRAM • FPGA Interface **PGAMOD** • 1x DSP • 4x XC4000 • FPGA Interface	• Compiled DSP code or FPGA functions can directly be called from host software • Parameters can be passed or functions can be called that exist in DSPs or FPGAs from C • FPGA function can be created in Giga C, Orcad, VHDL or XILINX tools



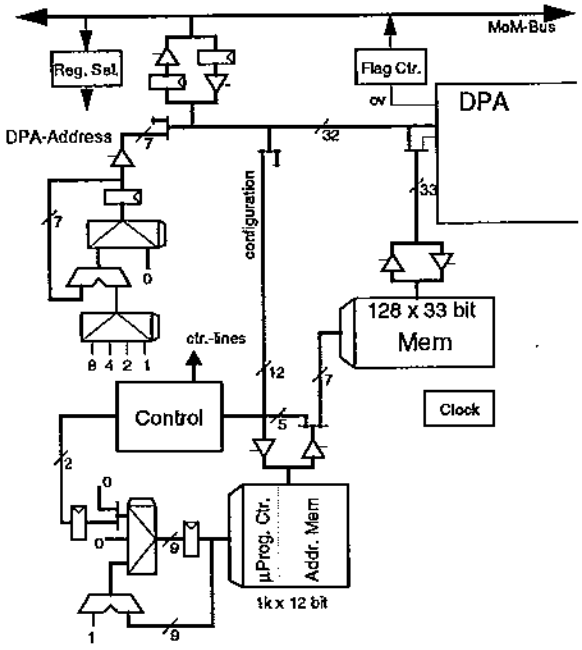
• FPGA-rALU

- XILINX XC 4008
- Floating-point ALU, Mult./Div.
- variable reconfigurable clock

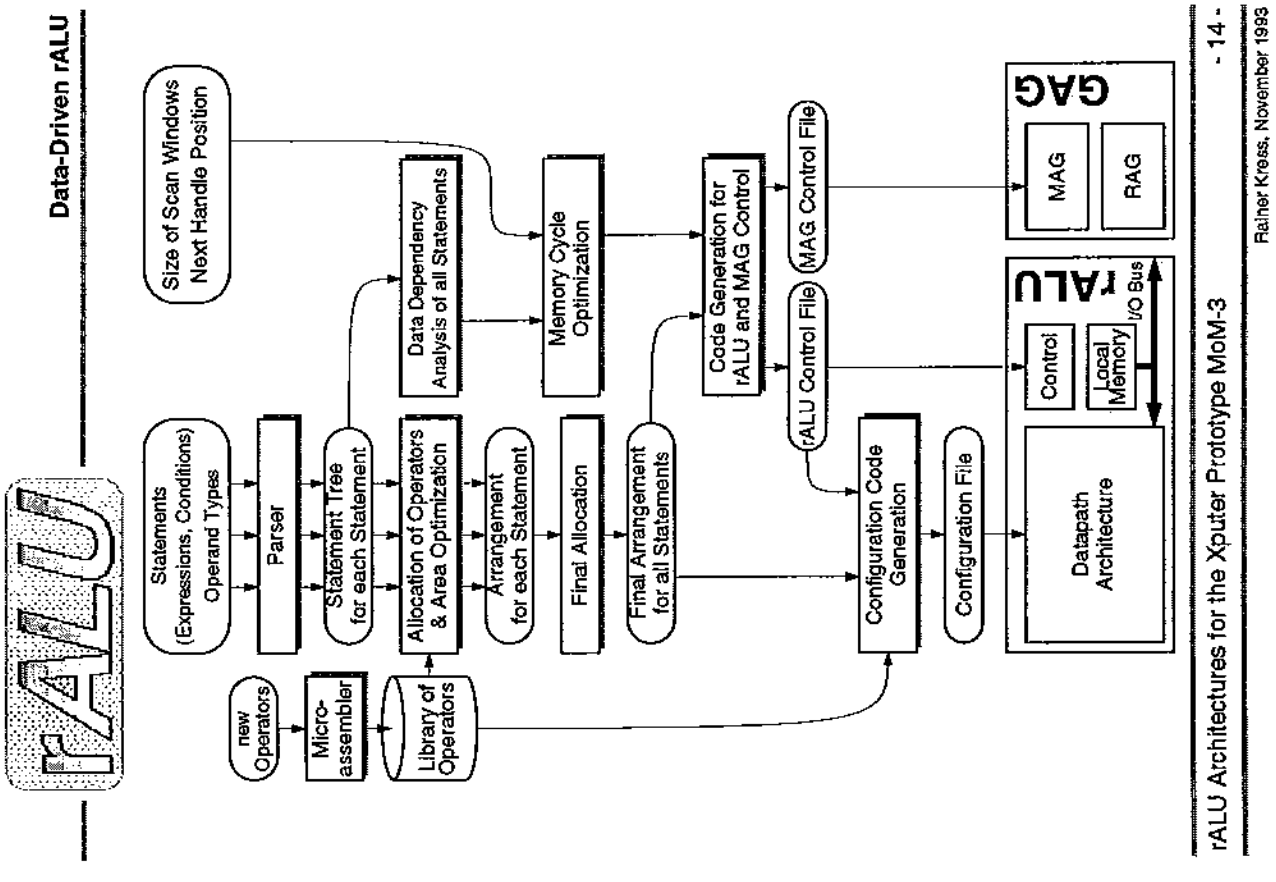


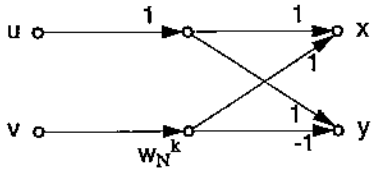
• Reconfigurable Datapath Architecture (DPA)

- 32 bit datapath
- partial reconfigurability
- scalable to arbitrarily large arrays
- rapid prototyping of high speed datapaths
- data-driven reconfigurable ALU for Xputers

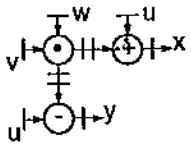


- **Control for the Reconfigurable Datapath Architecture (DPA)**
- 128 x 33 bit internal memory as register file
- micro-programmable control

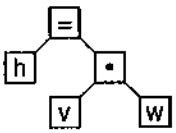


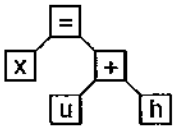


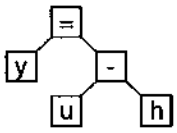
FFT-Butterfly Example

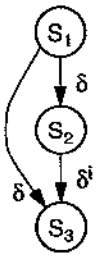


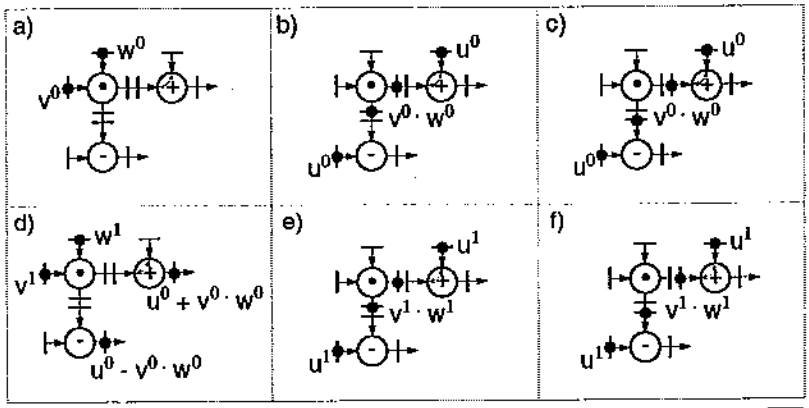

```
scan_window (1, 5);          (1)
next_handle (1, 0);          (2)
in  real  u, v, w;           (3)
out real  x, y;              (4)
aux_var h;                   (5)
h = v * w;      //S1        (6)
x = u + h;      //S2        (7)
y = u - h;      //S3        (8)
end;                         (9)
```





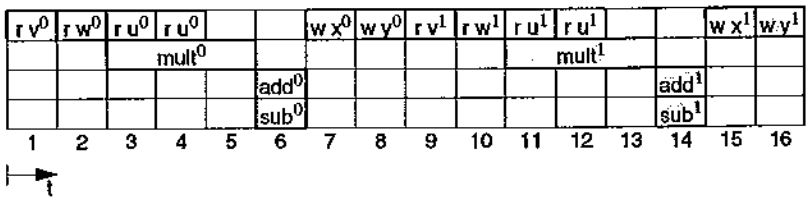




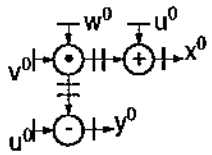


control code for the memory address generator

```
START: waitaddr;                (1)
       read (0, 1);   //v       (2)
       read (0, 0);   //w       (3)
       read (0, 2);   //u       (4)
       waitralu;                (5)
       write (0, 4);   //x      (6)
       write (0, 3);   //y      (7)
       goto START;              (8)
```

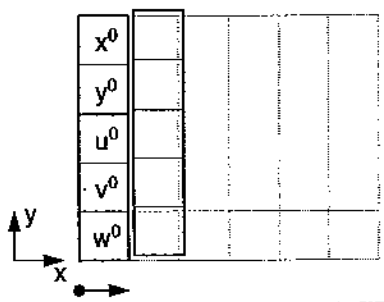


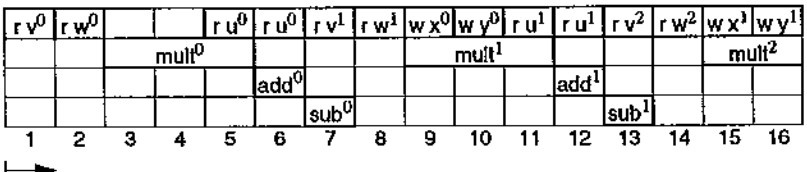
normal parallel execution

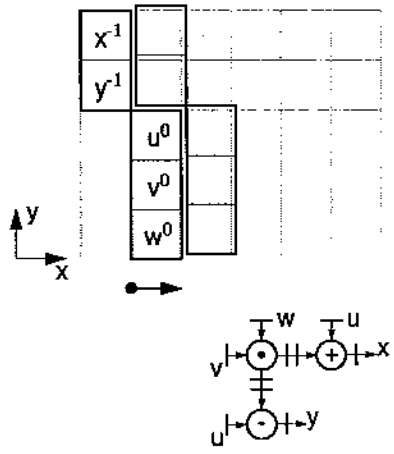


control code for the DPA-rALU

Instruction	Comm.	Addr. DPA	int. Mem. Addr. read	write	
MoM-Bus -> DPA_mAck	v	0	-	-	(1)
MoM-Bus -> DPA_mAck	w	1	-	-	(2)
MoM-Bus -> DPA_oAck	u	2	-	-	(3)
MoM-Bus -> DPA_mAck	u	3	-	-	(4)
DPA -> MoM-Bus	x	4	-	-	(5)
DPA -> MoM-Bus	y	5	-	-	(6)
end	-	-	-	-	(7)







pipelined execution

control code for the memory address generator

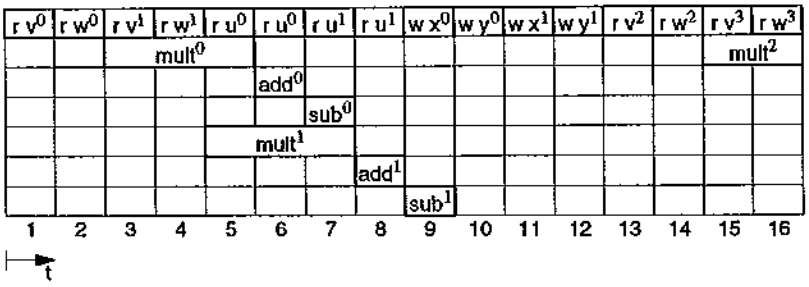
```
START:  waitaddr;                          (1)
        read (1, 1);    //v(0)             (2)
        read (1, 0);    //w(0)             (3)
        waitralu;                          (4)
SHORT:  read (1, 2);    //u(n-1)           (5)
        ifend FWRITE;                      (6)
        waitaddr;                          (7)
        read (1, 1);    //v(n)             (8)
        read (1, 0);    //w(n)             (9)
        waitralu;                          (10)
        write (0, 4);    //x(n-1)          (11)
        write (0, 3);    //y(n-1)          (12)
        goto SHORT;                        (13)
FWRITE: shift;                             (14)
        write (0, 4);    //x(N);           (15)
        write (0, 3);    //y(N);           (16)
        goto START;                        (17)
```

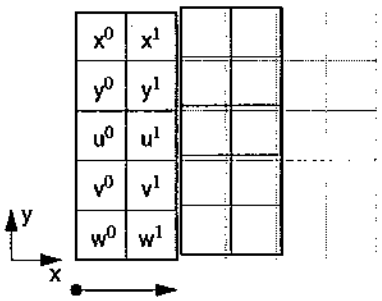

pipelined execution

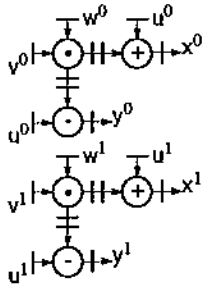
control code for the DPA-rALU

	Instruction	Comm.	Addr. DPA	int. Mem. Addr. read	write	
	MoM-Bus -> DPA_mAck	v	0	-	-	(1)
	MoM-Bus -> DPA_mAck	w	1	-	-	(2)
	Addr_incr_two	-	-	-	-	(3)
	MoM-Bus -> DPA_oAck	u	4	-	-	(4)
	MoM-Bus -> DPA_mAck	u	5	-	-	(5)
LOOPST:	MoM-Bus -> DPA_mAck	v	0	-	-	(6)
	MoM-Bus -> DPA_mAck	w	1	-	-	(7)
	DPA -> MoM-Bus	x	2	-	-	(8)
	DPA -> MoM-Bus	y	3	-	-	(9)
	MoM-Bus -> DPA_oAck	u	4	-	-	(10)
	MoM-Bus -> DPA_mAck	u	5	-	-	(11)
	ifshift goto LOOPST		-	-	-	(12)
	Addr_incr_two	-	-	-	-	(13)
	DPA -> MoM-Bus	x	2	-	-	(14)
	DPA -> MoM-Bus	y	3	-	-	(15)
	end	-	-	-	-	(16)

parallel execution with more hardware resource

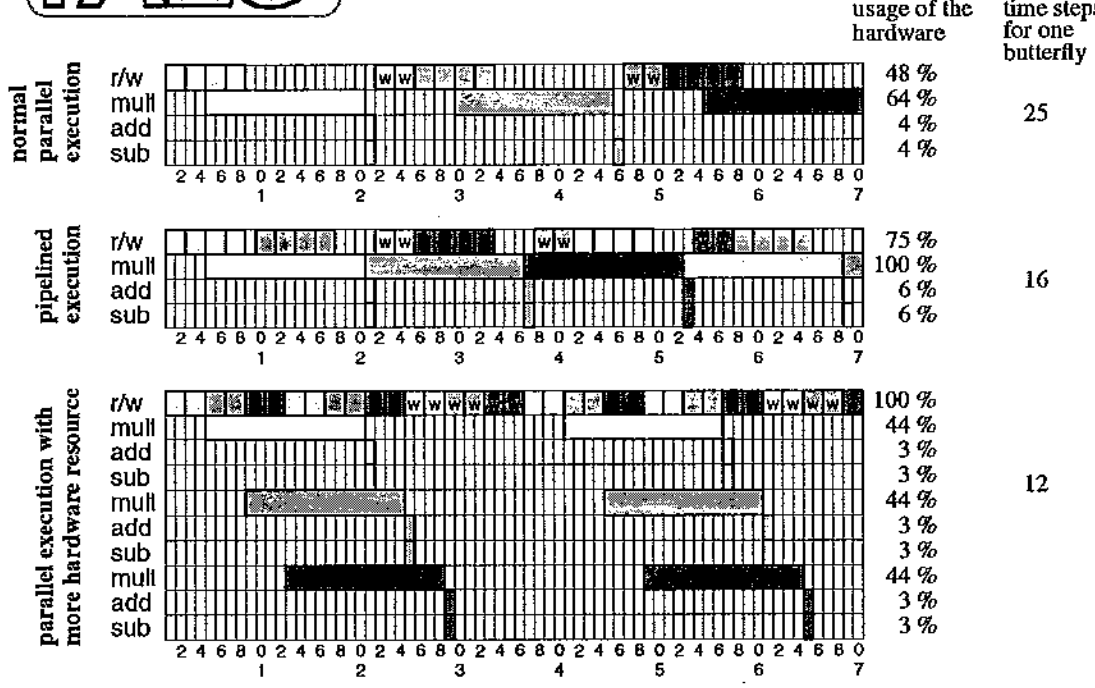






control code for the memory address generator

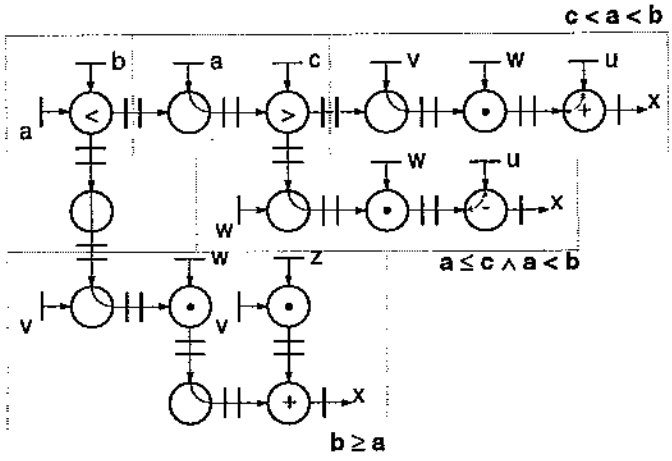
```
START: waitaddr;                          (1)
       read (0, 1);     //v(0)           (2)
       read (0, 0);     //w(0)           (3)
       read (1, 1);     //v(1)           (4)
       read (1, 0);     //w(1)           (5)
       read (0, 2);     //u(0)           (6)
       read (1, 2);     //u(1)           (7)
       waitralu;                          (8)
       write (0, 4);     //x(0)          (9)
       write (0, 3);     //y(0)          (10)
       write (1, 4);     //x(1)          (11)
       write (1, 3);     //y(1)          (12)
       goto START;                        (13)
```

Conditions

Example:

```
if (a < b)                    (1)
  if (a > c)                  (2)
    x = u + v * w;            (3)
  else                        (4)
    x = u - w * w;            (5)
else                          (6)
  x = v * w + y * z;          (7)
```



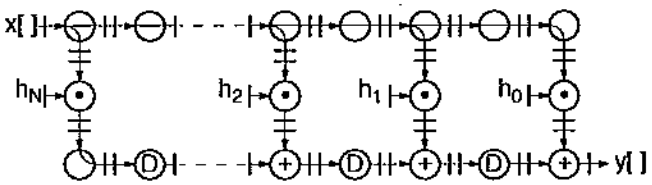
selection with condition bit

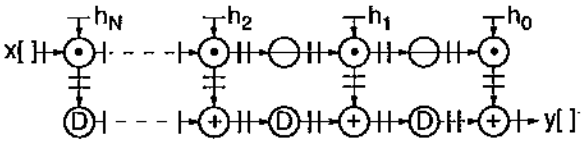
- **condition bit equal zero: no evaluation, result not valid**
- **condition bit equal one: evaluation, result valid**

Delay Operator

Finite Impulse Response Filter

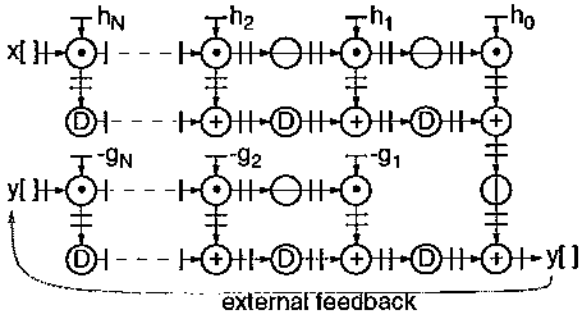
$$y_N = \sum_{k=0}^{N} h_k x_{N-k}$$





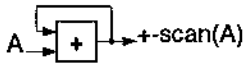
Infinite Impulse Response Filter

$$y_N = \sum_{k=0}^{N} h_k x_{N-k} - \sum_{k=1}^{N} g_k y_{N-k}$$



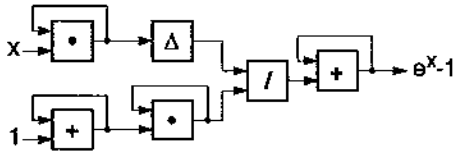
Parallel Prefix or Scan Operators

e. g. *add-scan*



A[ ] = {1, 4, 7,. 2, 6, 0, 3}

+-scan(A) = {1, 5, 12, 14, 20, 20, 23}

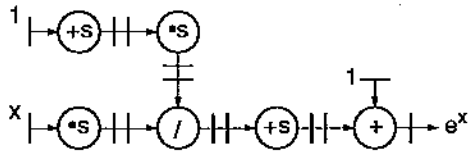


other scan primitives, e. g.:
max-scan, min-scan, and-scan, etc.

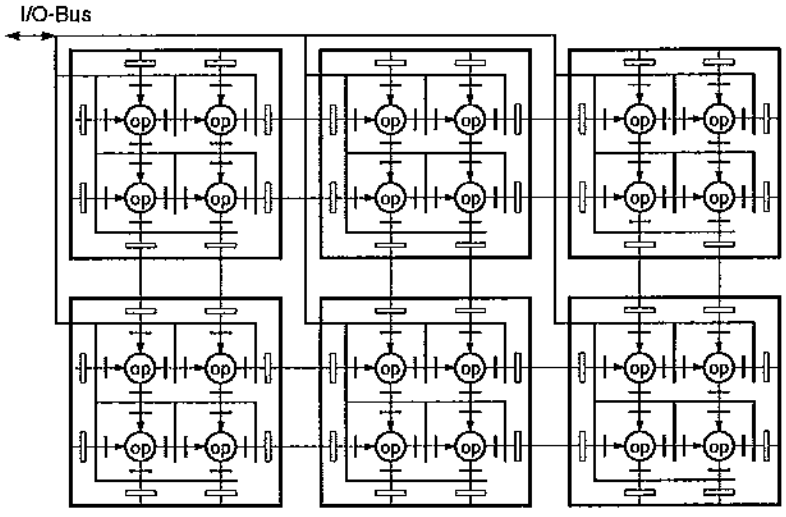
example:

$$e^x = \sum_{n=0}^{N} \frac{x^n}{n!} \qquad \text{for } |x| < \infty$$

$$= 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \ldots$$



or
Hardware Bubble-Sort with *max-scan*



• DPA-rALU implemented in FPGAs

- variable bit datapath
- operators can be implemented in parallel if necessary
- programmable by standard synthesis tools
- one FPGA for control and register file

- **rALU Emulator**
  - sequential emulation with optimized loading strategy for horizontal and vertical overlapping scan windows
- **FPGA based rALU**
  - any algorithm can be implemented
  - good for bit level operations and pattern matching
- **Data-Driven rALU**
  - parallel and pipelined evaluation of statements possible
  - optimized loading strategy for all overlapping scan windows
  - fast and automatic development system will be available
  - concept can be implemented in FPGAs
    ➥ operator optimization possible